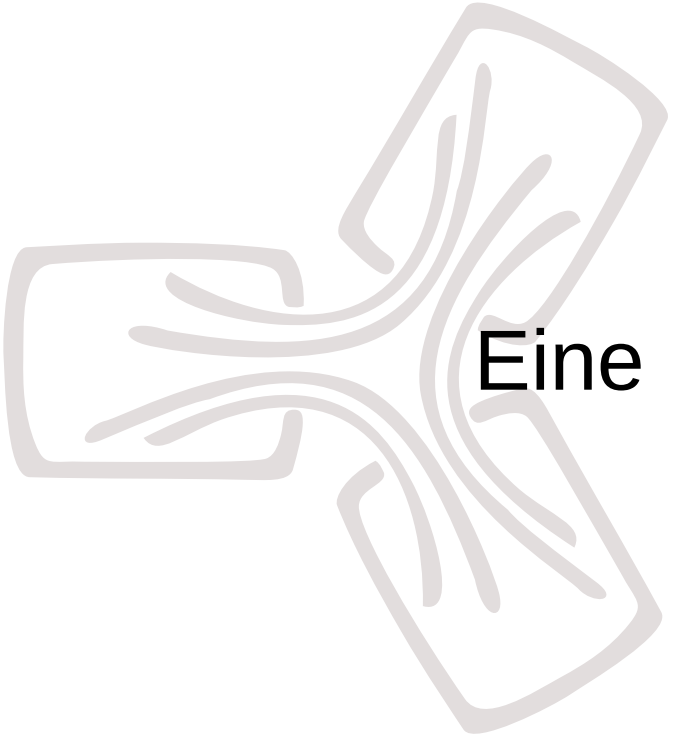
Regelungstechnik



Eine kurze Einführung

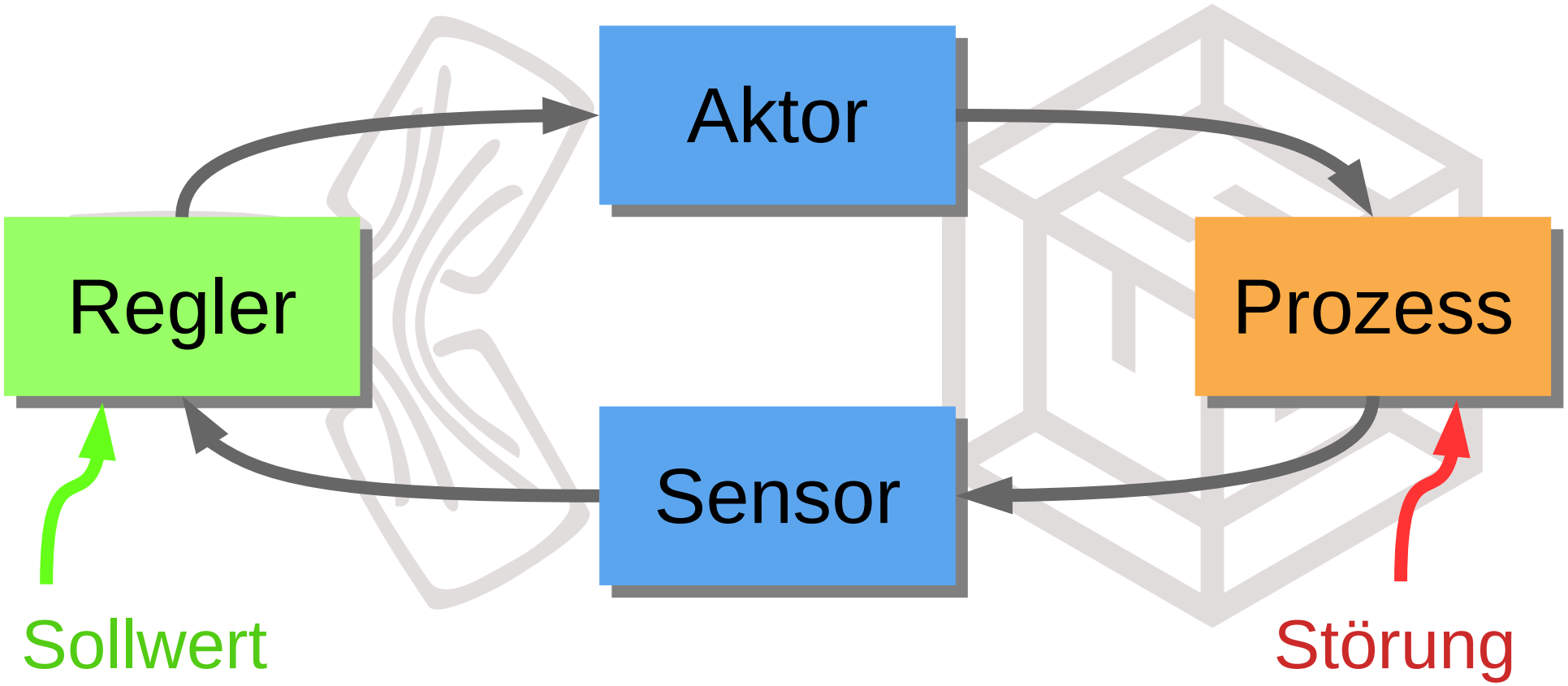
Regelungstechnik

- Übersicht und Begriffe
- Zweipunkt-Regler
- PID-Regler
- Weitergehende Konzepte
- Praktische Umsetzung
- Simulation

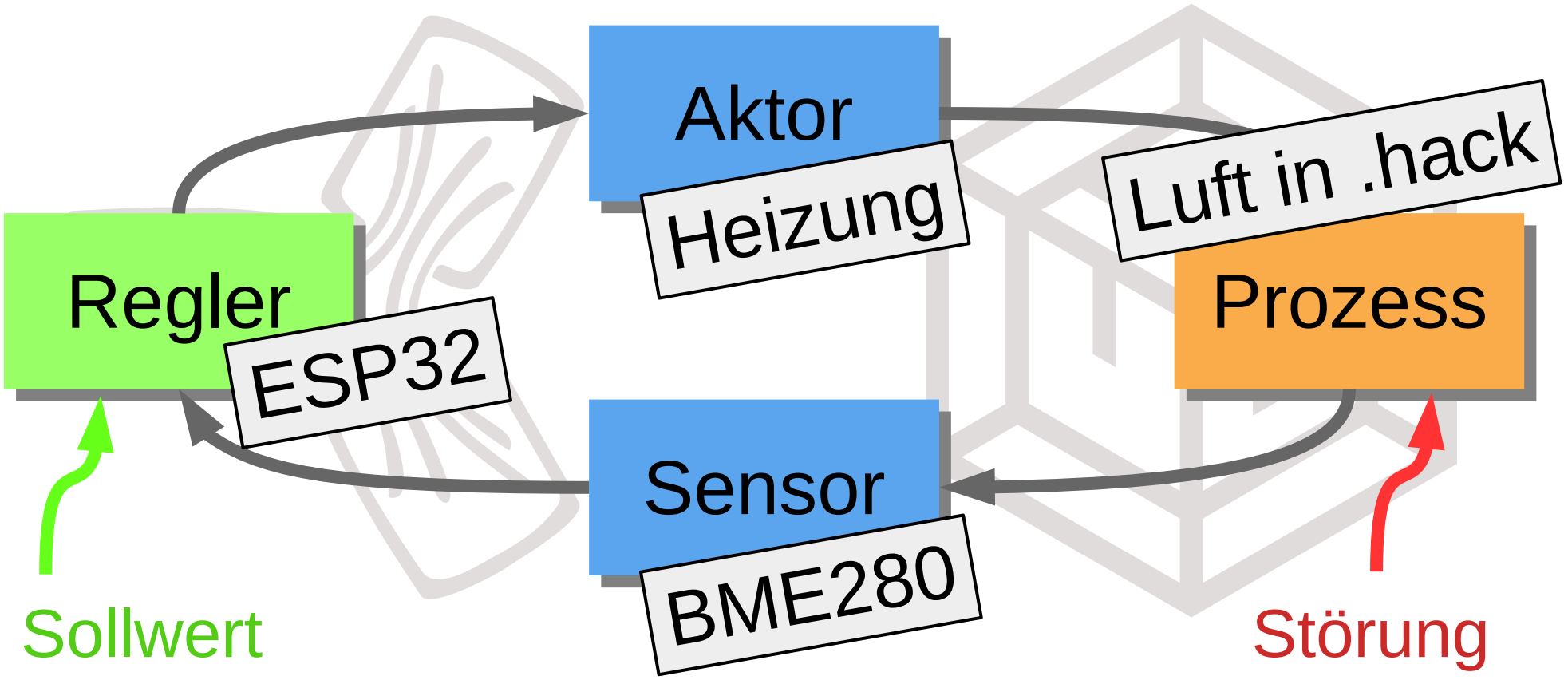
Regelung vs. Steuerung

- Wert einstellen, z.B. Temperatur, Position
- Steuerung: Signale anhand von Annahmen
  - Schrittmotor: Achse bewegt sich 0.25mm pro Schritt
  - Keine Korrektur bei Abweichung (ggf. manuell)
- Regelung: Messen und Abweichung korrigieren
  - Positionsregelung beim Servomotor
  - Temperaturregelung beim Herd

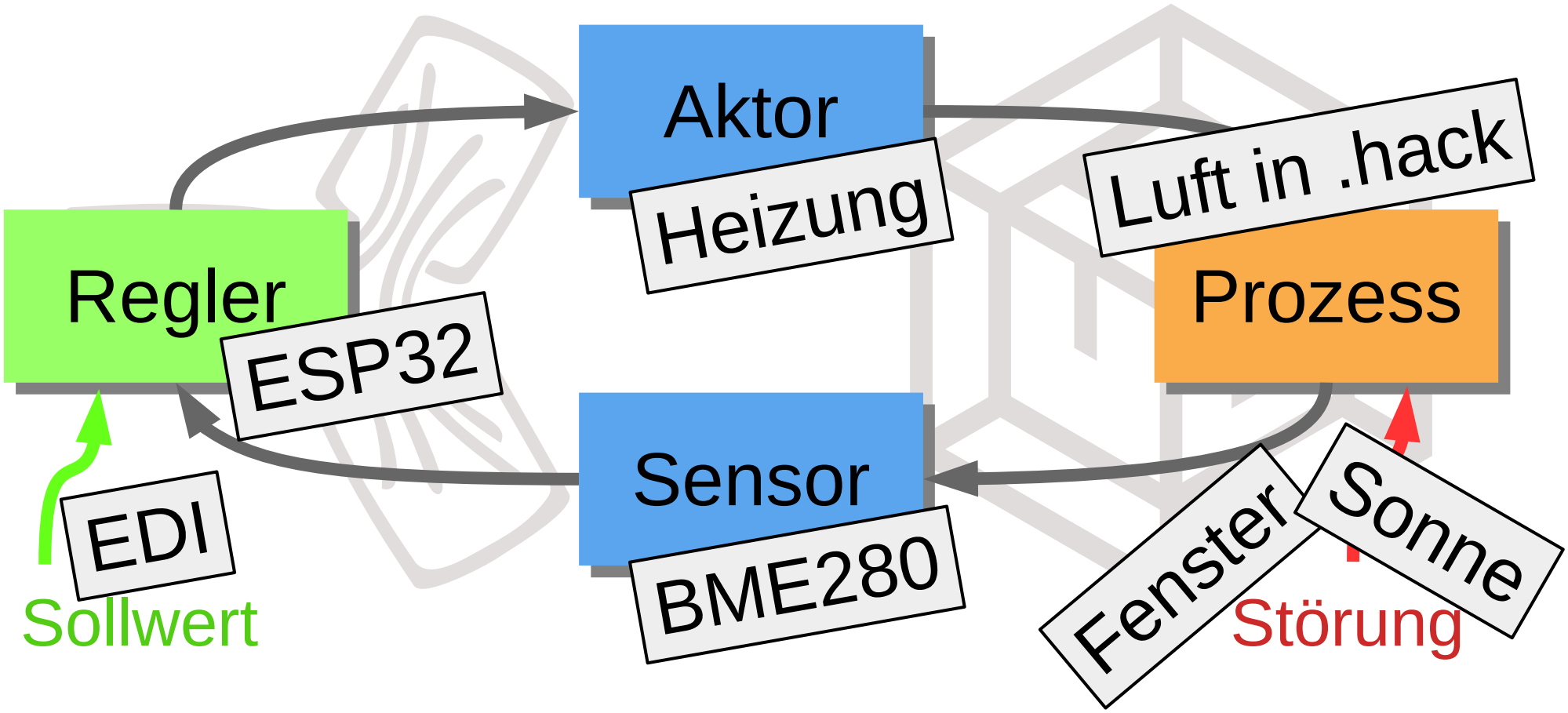
Regelkreis



Regelkreis

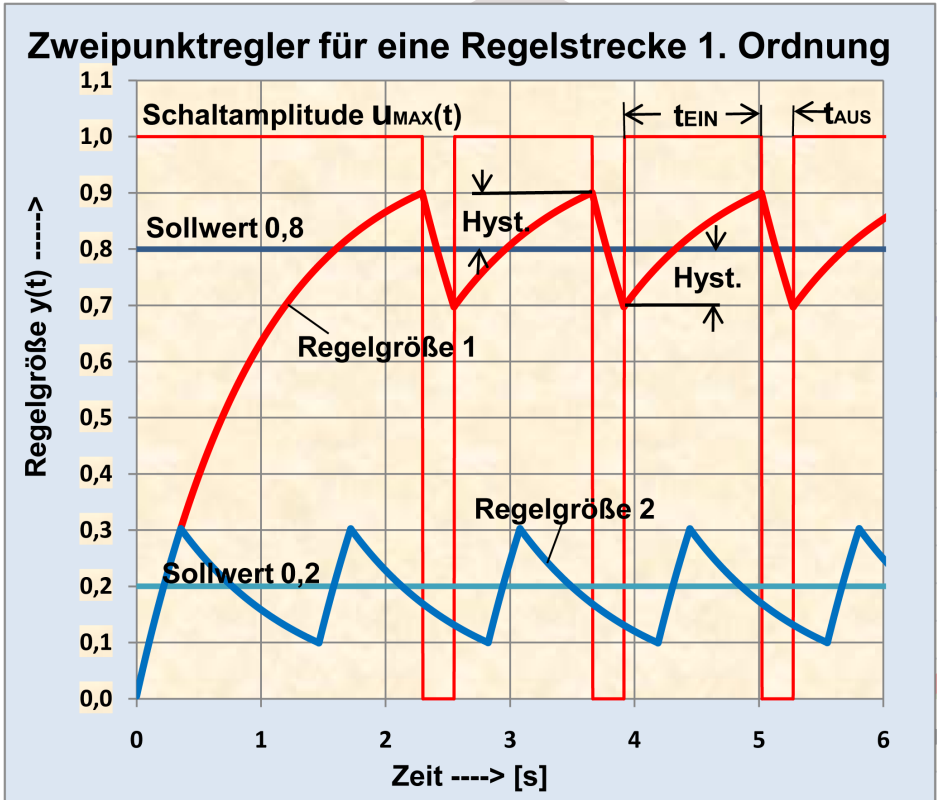

Aktor
Heizung
Luft in .hack
Regler
ESP32
Prozess
Sensor
BME280
Sollwert
Störung

Regelkreis


Aktor
Heizung
Luft in .hack
Regler
ESP32
Prozess
Sensor
BME280
EDI
Sollwert
Fenster
Sonne
Störung

Zweipunkt-Regler

- Herdplatte: klack-klack -> an/aus
- Hysterese!
- Für träge Strecke mit geringen Anforderungen



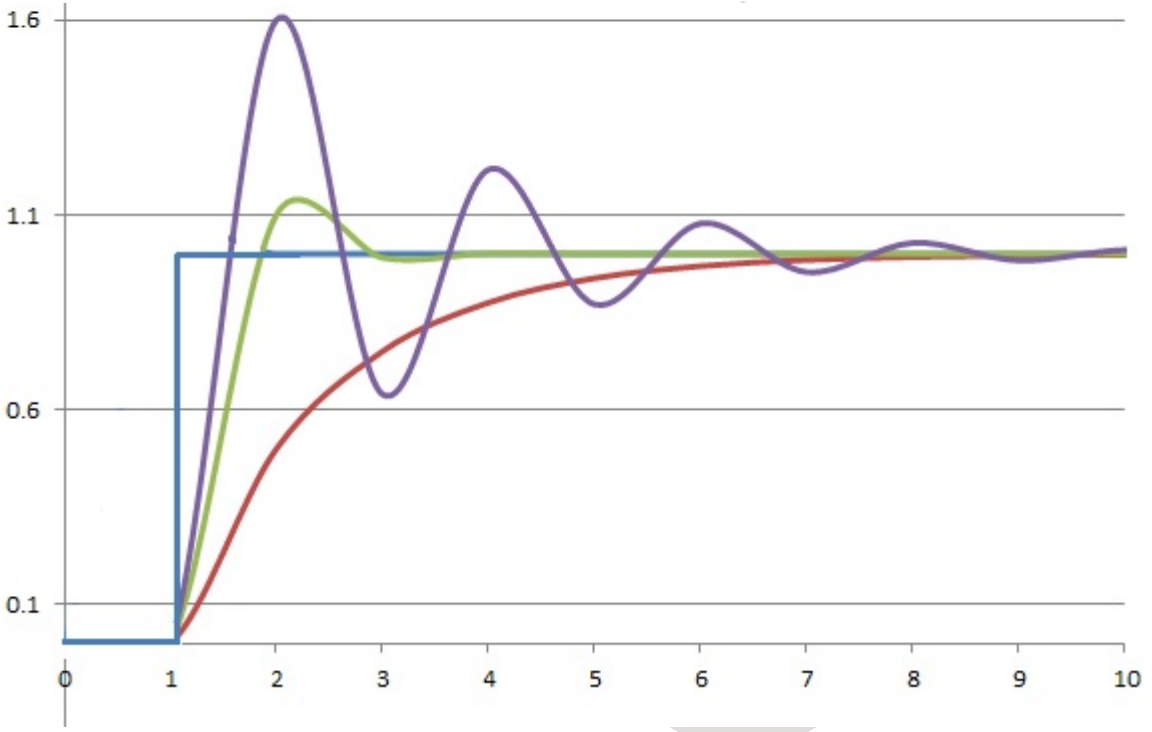
https://de.wikipedia.org/wiki/Datei:Zweipunktregler_f%C3%BCr_eine_regelstrecke_erster_ordnung.png

PID-Regler

- „Standard-Regler“
- Grundgedanke: schwächerer Regeleingriff, wenn nah am Sollwert
- Benannt nach den drei Teilen: P-, I-, D-Anteil
- Teile können weggelassen werden, z.B. PI-Regler

P-Anteil

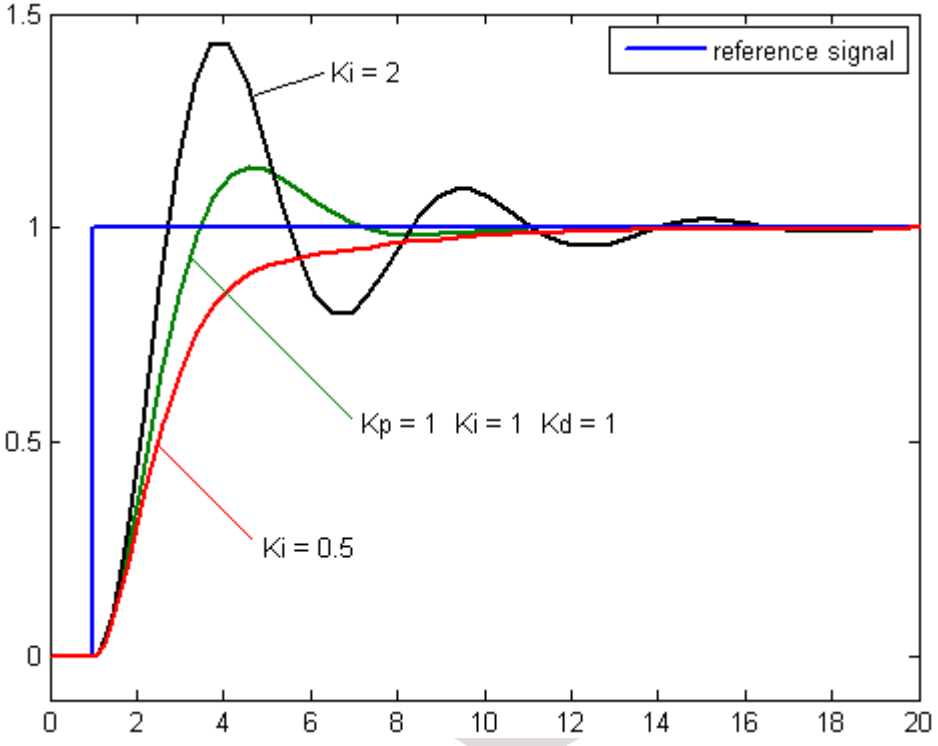
- Proportional zur Abweichung:
  $P = k_p * e(t)$
- Verhalten abhängig von Parameter k_p



https://en.wikipedia.org/wiki/PID_controller#Proportional_term

I-Anteil

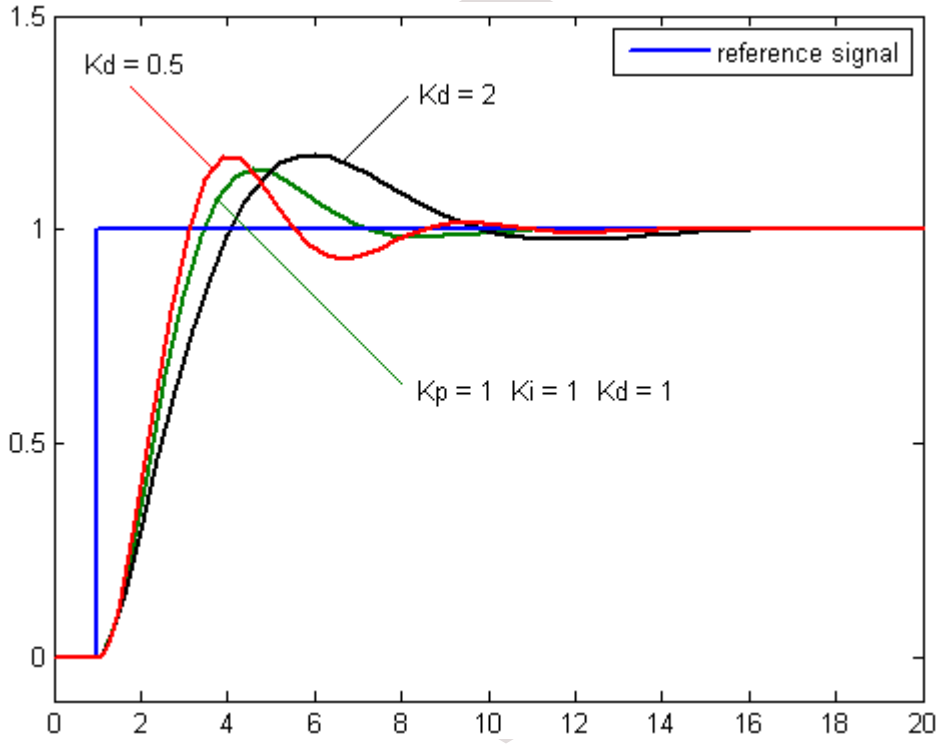
- Integral über Abweichung:
  I = kI * integral(e(t))
- Je länger der Fehler bleibt, desto mehr
- Beseitigt Restfehler
- Oft in Strecke enthalten



https://en.wikipedia.org/wiki/PID_controller#Integral_term

D-Anteil

- Ableitung der Abweichung:
  $D = k_D * d/dt\ e(t)$
- Schnelle Reaktion auf Änderungen
- Erlaubt größeres k_P



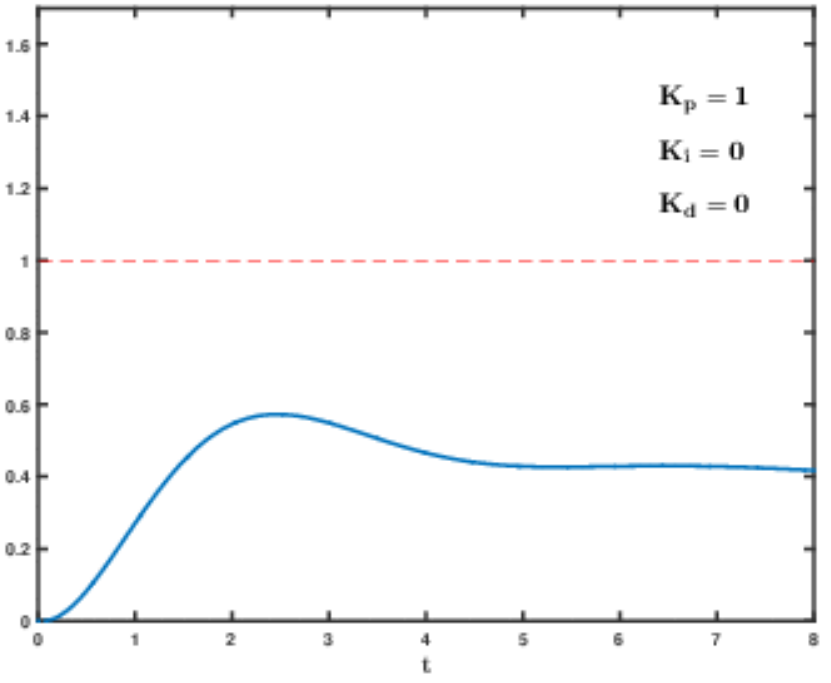
https://en.wikipedia.org/wiki/PID_controller#Derivative_term

D-Anteil

- Problem: anfällig für Rauschen
- Überschlagsrechnung – womöglich Quatsch:
  - 50 Hz Rauschen mit 0.1 Kelvin:
    0.1 K * 2*50 Hz = 10 K/s
  - Fenster auf mit 10 K in 4 sec:
    10 K / (4 sec) = 2.5 K/s
- Fazit: ausprobieren

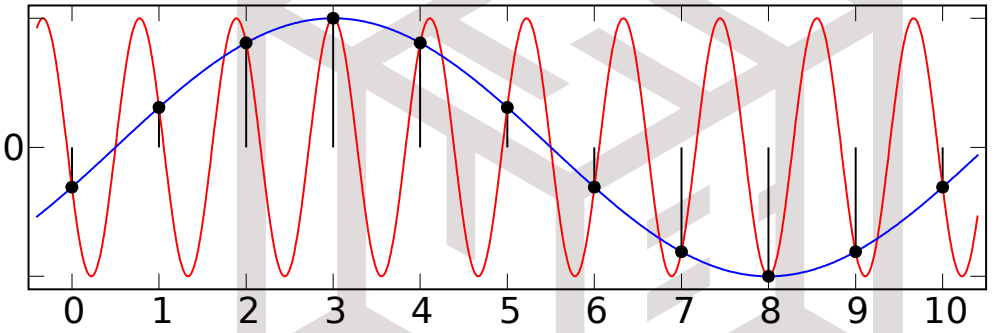
Parameter einstellen

- Diverse Verfahren, aber oft mit math. Modell
- Wenn manuell gut genug: Erst P, dann I, dann D
- Andere siehe Wikipedia



Praktische Umsetzung

- diskrete Regelung: Wert und Zeit
  - theoretisch schlechter
  - egal, wenn fein genug
  - Gleichmäßige Zeitschritte!
- Aliasing: hohe Freq. analog filtern!
- Verzögerung klein halten



https://en.wikipedia.org/wiki/Aliasing#Sampling_sinusoidal_functions

Praktische Umsetzung

- Beispiel-Code:
  - error = set_point – value
  - sum_error += error
  - diff_error = last_error – error; last_error = error
  - output = kP * error + kI * sum_error + kD * diff_error
- App Note von Atmel: AVR221, längeres Beispiel

Anti-Windup

- Sättigung, z.B. seitlicher Anschlag bei Positionsregelung
- Auswirkung: Fehler bleibt besten, I-Anteil wächst immer weiter -> Regler macht Quatsch
- Lösung:
  - Einfach: Grenzen für I-Anteil
  - Besser: Sättigung erkennen, Integral festhalten

Verbesserungen

- Strecke verbessern oft am besten:
  - Mehr Sensoren oder geschickter positionieren
  - Verzögerung reduzieren, z.B. Lüfter verteilt warme Luft schneller
  - Sprünge vermeiden, z.B. Spiel im Getriebe
- Verhalten der Strecke intelligent kompensieren
- Kaskadenregelung: Strom -> Drehzahl -> Lage

Strecken-Modelle

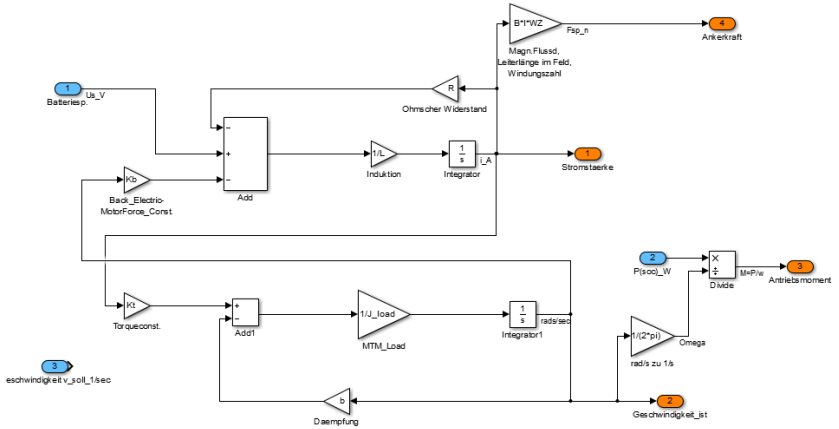
- Bausteine: Tiefpass vs. Totzeit, Integral
- Beispiel: Temperatur-Steuerung
  - Wetterbericht / Außentemperatur
  - Temperatur/Energiegehalt der Raumluft abschätzen: Wenn man gerade geheizt hat, ist die Temperatur ungleichmäßig verteilt.

Simulation

- Lineare Differentialgleichungen für viele Bausteine: Tiefpass, Motor, Raumluft
- Nicht-lineare Anteile, z.B. Schalter, Sättigung
- Grafisch zusammenklicken, nummerisch simulieren
- Matlab/Simulink oder Xcos/Scilab

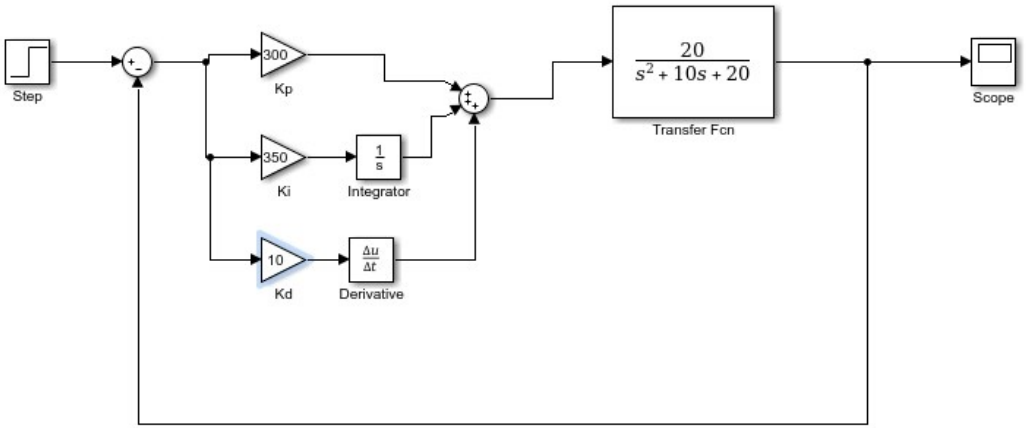
Simulation

DC-Motor



https://www.mikrocontroller.net/topic/326266

PID-Regler



https://microcontrollerslab.com/pid-controller-design-simulink/